



PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09305440 A**

(43) Date of publication of application: 28 . 11 . 97

(51) Int. Cl.

G06F 11/30**G06F 11/00****G06F 13/00**(21) Application number: **08115984**

(22) Date of filing: 10 . 05 . 96

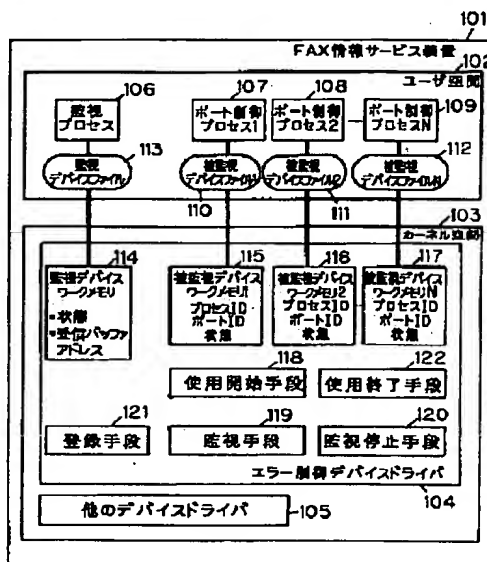
(71) Applicant: **MATSUSHITA GRAPHIC COMMUN
SYST INC**(72) Inventor: **SUZUKI TAKU****(54) ERROR PROCESSING METHOD**

(57) Abstract:

PROBLEM TO BE SOLVED: To simplify an error process, to mitigate the load of the error process, and to evade a state wherein a device can not be controlled owing to the abnormal end of an operation management process in a processor consisting of plural operating on a multitask monitor and an operating system.

SOLUTION: This method has a monitor process which monitors an error and processes to be monitored; if an error occurs, information to be reported from a monitored process to the monitoring process is previously held before the error occurrence to the monitored process and reported to the monitoring process 106 after the error occurrence to the monitored process. If the abnormal end of the monitoring process 106 is detected during the error monitoring by the monitoring process 108, the system is reset and the monitoring process 106 is started again.

COPYRIGHT: (C)1997,JPO



THIS PAGE BLANK (USPTO)

(19)日本国特許庁 (JP)

(11)特許出國公團番号

特開平9-305440

(43)公開日 平成9年(1997)11月28日

(51)Int.Cl.	識別記号	序内整理番号	PI	技術表示箇所
G 0 6 F 11/30			G 0 6 F 11/30	K
11/00	3 1 0		11/00	3 1 0 C
13/00	3 6 1		13/00	3 5 1 M

審査請求 未請求 請求項の数10 OL (全 12 頁)

(21)出國證明書 特國平8-115984

(71) 出票人 000187736

(22) 出願日 平成8年(1996)5月10日

東京都目黒区下目黒2丁目3番8号

(72) 兎明者 鈴木 卓

東京都目黒区下目黒2丁目3番8号 松下

電送株式会社内

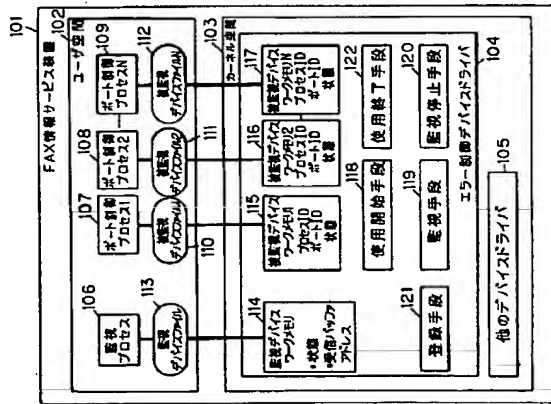
(74)代理人 弁理士 滝本 智之 (外1名)

(54) 【発明の名称】 エラー処理方法

(57)【要約】

【課題】 本発明は、マルチタスクモニタやオペレーティングシステム上で動作する複数のプロセスからなる処理装置において、エラー処理の階層化、エラー処理による負荷の軽減と運用管理プロセスの異常終了によって装置が制御不能になる状況回避することを目的としてい

【解決手段】 エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記被監視プロセスから前記監視プロセスに通知すべき情報を前記被監視プロセスでのエラー発生前に予め保持し、前記監視プロセスでエラー発生後に前記情報に基づき前記監視プロセスに通知し、精成とした。また、監視プロセスによるエラー監視実行中に監視プロセスの異常終了が検出された場合には、システムリセットを実行し再び監視プロセスを起動する精成とした。



復帰する(ステップ901)。

【0009】一方、エラー監視の対象である、被エラー監視プロセスの処理は、以下のとおりである。図10において、被エラー監視プロセスはイベントの受信待ちの状態待機し(ステップ1001)、イベントが受信されると、そのイベントがエラー監視プロセスからの状態チェックメッセージが否かをチェックする(ステップ1002)。

【0010】発生したイベントが、エラー監視プロセスからの状態チェックメッセージでない場合には、通常のイベント処理を実行し(ステップ1003)、更に、その通常のイベント処理において障害が発生しているかをチェックし(ステップ1004)、正常であればイベント受信待ちの状態に復帰する(ステップ1005)。障害が検知された場合には、監視プロセスに対してエラー通知メッセージを送信し(ステップ1006)、イベント受信待ちの状態に復帰する(ステップ1001)。

【0011】逆に、発生したイベントが、エラー監視プロセスからの状態チェックメッセージである場合には、被エラー監視プロセスはその状態情報を監視プロセスに送信する処理を行う(ステップ1007)。

【0012】この際、エラー監視プロセスが、定期的にエラー監視の対象となる全てのプロセスに対して状態チェックメッセージを送信し、これを受信した被エラー監視プロセスが状態情報を監視プロセスに送信するような監視形態とする理由は、例えば、被エラー監視プロセスのプログラムの不具合等により、エラーの検出自体を行なえず通知することができないという事態を回避するためである。

【0013】**【発明が解決しようとする課題】**しかし、上述の従来技術の構成では、エラーを検出するために監視対象となるプロセスと監視プロセスとがメッセージを交換する必要があるため、本来の処理のほかにプロセス間通信処理部やタイマー処理を組み込まなければならず、ソフトウェアの処理が複雑になる。

【0014】また、監視プロセスが定期的に全ての監視対象プロセスに対して状態チェックメッセージを送信するため、特に監視対象数が多い場合は計算資源が無駄に使用されることとなる。

【0015】また、装置管理機能を持つ監視プロセスが異常終了した場合に、他のプロセスが監視プロセスの異常終了を検知し、エラー処理を行わない限り、装置全体が制御不能になるという問題がある。この問題に対して、監視プロセスの障害検知をメッセージによる通知方式で行う場合には、各監視対象プロセスが逆に監視プロセスに対して状態チェックメッセージを送信することに

なり、ソフトウェアがますます複雑化し、負荷もさらに大きくなることとなる。

【0016】本発明は、上述の問題に鑑みて為されたもので、マルチタスクモディヤオペレーティングシステム上で動作する複数のプロセスからなる処理装置において、エラー処理の簡潔化、エラー処理による負荷の軽減と運用管理プロセスの異常終了によって装置が制御不能になる状況を回避すること、を目的としている。

【0017】**【課題を解決するための手段】**上述の問題を解決するため、本発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記監視プロセスから前記監視プロセスに通知すべき情報を前記監視プロセスでエラー発生前に予め保持し、前記監視プロセスでエラー発生後に前記情報を前記監視プロセスに通知する構成とした。また、監視プロセスによるエラー監視実行中に監視プロセスの異常終了が検出された場合には、システムリセットを実行し再び監視プロセスを起動する構成とした。

【0018】この構成により、監視プロセスが定期的に被監視プロセスの状態をチェックする必要がなくなり、エラー処理部のプログラム量を限定的に減らすことが可能になり、また、エラー検知にプロセス間通信を使用しないので装置の計算資源の浪費を避けることができる。また、運用管理プロセスの異常終了によって装置が制御不能になる状況を回避することができる。

【0019】**【発明の実施の形態】**請求項1記載の発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記監視プロセスから前記監視プロセスに通知すべきエラー情報を前記監視プロセスでエラー発生前に予め保持し、前記監視プロセスでエラー発生後に前記情報を前記監視プロセスに通知するものであり、エラー検出に必要な情報を被監視プロセスとは異なる場所に保持しておき事後に監視プロセスに通知することにより被監視プロセスが異常終了した場合でも確実にエラー検出を行ない得る。

【0020】請求項2記載の発明は、請求項1において、システム起動時に呼出されるエラー情報を保持する領域を導込み可能な状態にする使用開始手段を有するものであり、これによりシステム起動時に際しては確実にエラーチェックの準備が整うこととなる。

【0021】請求項3記載の発明は、請求項1において、エラー発生の際に前記監視プロセスから前記監視プロセスに通知すべきエラー情報を保持する処理を被監視プロセス起動時に実行する登録手段を有するものであり、被監視プロセス起動の際にエラー通知に必要な情報の登録が実行されるため、被監視プロセス起動直後にそのプロセスのエラー監視が可能になる。

【0022】請求項4記載の発明は、請求項1において、監視プロセスにより起動された後命令受け待ちの状態待機し、エラー情報認識後にそのエラー情報を読み出して前記監視プロセスに通知する監視手段を有するものであり、監視手段がエラー検出を開始し、実行するため、監視手段を呼出だけで全ての監視対象プロセスで生じたエラーを検知することが可能となる。

【0023】請求項5記載の発明は、請求項4において、エラー発生による被監視プロセス終了時にエラー情報を監視手段に認識させる使用終了手段を有するものであり、監視手段を実行させることで、エラー通知に必要な情報を被監視プロセス終了時に確実に監視プロセスに通知することが可能となる。

【0024】請求項6記載の発明は、請求項4において、監視プロセスにより起動される監視手段に対して監視停止命令を通知して監視を終了させる監視停止手段を有するものであり、監視手段によりエラー監視を開始し、監視終了手段によりエラー監視を中断でき、監視不要な処理に対してのエラー監視、エラー通知を避けることが可能となる。

【0025】請求項7記載の発明は、請求項5において、監視プロセスが通常のイベント処理を実行するステップとエラー監視処理を実行するステップとを有して構成されるものであり、監視プロセス106がエラー監視をする処理としない処理を明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を実行する必要がなくなる。

【0026】請求項8記載の発明は、エラー監視プロセスと、複数の被監視プロセスと、を有し、この複数の被監視プロセスの異常終了状態を各々検出し、前記監視プロセスが異常終了した場合に起動し異常終了した被監視プロセスを特定する情報を前記エラー監視プロセスに通知するエラー処理を実行するものであり、被監視プロセスが異常終了した場合でも確実にエラー検出、エラー通知を行ない得る。

【0027】請求項9記載の発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、これらのプロセスに対応して各々設けられプロセス実行開始時にオート起動となりプロセス実行停止時にクロース状態となるデバイスドライバと、を有し、前記いずれかのプロセスが異常終了しデバイスドライバがクロース状態となった場合にその異常プロセスを特定する情報を前記監視プロセスに通知するものであり、被監視プロセスの状態監視用のデバイスドライバによりエラー検知を確実に行ない得る。

【0028】請求項10記載の発明は、請求項8又は請求項9において、監視プロセスによるエラー監視実行中にエラー制御手段が監視プロセスの異常終了を検出した場合には、システムリセットを実行し再び監視プロセスを起動するものであり、監視プロセスが監視中に異常終了した場合に装置全体が再起動するので、必要となるプロセスを全て再生でき、監視プロセスがなく装置の操作が不能になるという状況をなくすることが可能である。

【0029】以下、本発明の実施の形態を図面を参照して具体的に説明する。図1は、本発明のエラー制御デバイスを使用したFAX情報サービス装置のソフトウェア構成を示している。101はFAX情報サービス装置であり、このFAX情報サービス装置101のメモリ空間は、後に詳述するように、複数のプロセスが実行されるユーザ空間102と、基本オペレーティングシステム(以下、OSという)が実行される基本OS空間103と、から構成されている。更に、この基本OS空間103では、エラー制御デバイスドライバ104とそれ以外のデバイスドライバ105、例えば、複数チャネルフックシミュレーションポートドライバ等との双方が動作し、ユーザ空間102では、監視プロセス106とポート制御プロセス(107~109)との双方が動作するようになっている。

【0030】さて、FAX情報サービス装置101は、電話回線と1対1に対応するN個のポート制御プロセス(107から109)を有し、複数のフックシミュレーション情報を複数の情報ボックス(図示せず)に分類して蓄積して置ける。ポート制御プロセス(107から109)はユーザから来る呼の着信を待ち、着信後にユーザが所定の情報ボックスの情報の取出しの指示をした場合には、ポート制御等で受信し、指定された情報ボックスに蓄積されたフックシミュレーションを送信する制御を行う。ユーザが情報ボックスへの登録を指示した場合には、ポート制御プロセスはフックシミュレーションを受信し、指定の情報ボックスに蓄積する制御を行う。

【0031】このポート制御プロセス(107~109)が動作中にエラー発生を検出した場合には、各々のポート制御プロセス(107~109)に対応して設けられた被監視デバイスドライバ(110~112)にアクセスしてエラー通知を行なう。一方、監視プロセス106はFAX情報サービス装置の起動、停止処理を行う他に、監視デバイスドライバ110にアクセスしてポート制御プロセスの実行監視を行うようになっている。

【0032】エラー制御デバイスドライバ104は、基本OS空間103に、作業領域として監視デバイスドライバ110に対応する監視デバイスドライバ114と、被監視デバイスドライバ(110~112)に対応する被監視デバイスドライバメモリ(115~117)を有し、これらのワーキングメモリは、それぞれ対応するデバイスドライバの状態を記述する領域を有している。FAX情報サービス装置101が起動するときに、基本OSはエラー制御デバイス104の初期化処理を呼び出し、前記初期化処理はワーキングメモリ(114~117)を割り当て、各ワーキングメモリの状態フィールドをクロスに設定する。

【0033】また、エラー処理デバイスドライバ104

は、使用開始手段1118と、監視手段1119と、監視停止手段120と、登録手段121と使用終了手段122とを有しており、使用開始手段1118は対応するデバイスファイルがオープンされたときに基本OSにより呼び出され、使用終了手段122は対応するデバイスファイルがクローズされたときに基本OSにより呼び出されるプログラムであり、また、監視手段119と監視停止手段120と登録手段121とは、基本OSが提供するシステムコール（以下では入出力制御命令と呼ぶ）により実行される固有のプログラムである。

【0034】以上のようなソフトウェア構成を採るFA
X情報サービス装置におけるエラー制御デバイスドライ
バによるエラー処理フローを図2乃至図8を用いて説明
する。

【0035】最初に使用開始手段118について説明する。使用開始手段118は、エラー制御デバイスドライバの一部を構成するプログラムであり、システムが立ち上がり、いずれかのプロセスが開始されてデバイスフアイルがオープンされた時に基本OSにより呼出され、そのプロセスに対応するデバイスフアイルのワークメモリ(114~117)の状態フイールドをオープン状態を示しているものである。図2は、その動作の詳細フローを示している。

【0036】監視プロセス106またはポート制御プロセス（107～109）が、それぞれ、監視デバイスファイル113または被監視デバイスファイル（110～112）をオープンしたときに、基本OSはオープンされたそのデバイスファイルのパラメータとして、エラー制御デバイスドライバ104の使用開始手段118を呼び出す。

【0037】使用開始手段118は、パラメータで指定されたデバイスファイルが監視デバイスファイル113かどうかをチェックし（ステップ201）、監視デバイスファイル113であれば、既に監視デバイスファイル113がオープンされているかを監視デバイスワーケメモリ114の状態フィールドを参照することによりチェックする。その状態フィールドがオープンの状態であれば、既に他のプロセスが監視デバイスファイルを開いているということなので、ステップ203でエラーをリターンする。逆に、その状態フィールドの値がクローズであれば、ステップ204で状態フィールドをオープンに設定し、ステップ205にて成功をリターンする。

【0038】パラメータ指定のデバイスファイルが監視
デバイスファイル113でなければ、被監視デバイスフ
ァイル(110～112)であるので、ステップ206
において階層ファイルのデバイス番号をチェッ
クする。
デバイス番号が不正であればエラーをリターンするが
(ステップ208)、デバイス番号が正しければ、その
デバイスファイルに対応するワークメモリ(115～1

パラメータで指定されたデバイスファイルが監視デバイスファイル113であるか否かをチェックし（ステップ401）、監視デバイスファイル113でなければエラー処理とし（ステップ402）、監視デバイスファイル113であれば監視デバイスワークメモリ114の状態フィードバック値が監視中であるかチェックする（ステップ403）。状態フィードバック値が監視中でなければエラー処理とし（ステップ402）、監視中であれば監視デバイスワークメモリ114のバックアップアドレスフィールドで指定された受信バックアップに監視停止をライトし（ステップ404）、基本OSインターフェースであるWAKEUP処理をコールして（ステップ405）、受信待ちの状態になっている監視手段119を待ち状態から監視停止状態に渡り、成功をリターンする（ステップ307）。

【0045】以上一連の処理が監視手段119及び監視停止手段120の動作であり、エラー監視が必要な処理に対して監視手段119の起動をかけるだけで、エラーが検出されるか監視停止処理が実行されるまでエラー監視が執行されることとなる。また、監視停止処理も監視手段119とは独立したプログラムである監視停止手段120により実行される。

【0046】次に、監視手段121について説明する。監視手段121は、エラー制御デバイスドライバ104の一画を構成し、エラー監視の対象となるポート制御プロセス(107乃至109)の起動時に、各プロセスを特定するための情報の情報を基本OS空間上のワークメモリに登録したおためめのプログラムであり、図5は、その動作の詳細フローを示している。

【0047】ポート制御プロセス（107乃至109）がオープンされた被制御デバイスファイル（110～112）に対して出入力制御コールを実行し、登録手段を呼び出すと、基本OSは、被制御デバイスファイル（107乃至109）とポート制御プロセス（107乃至109）が提供するプロセスIDとポート番号を含む情報を受信する受信バッファのアドレスをパラメータとして登録手段121を呼び出す。登録手段121は、パラメータのデバイスファイルが被監視デバイスファイル（107乃至112）か否かをチェックし（ステップ501）、被監視デバイスファイル（110～112）でない場合はエラー処理をし（ステップ502）、被監視デバイスファイル（110～112）である場合は、そのデバイスファイルの番号をチェックする（ステップ503）。デバイスファイル番号が不正であればエラー処理をし（ステップ502）、デバイスファイル番号が正であれば、前記受信バッファの内容を対応する被監視デバイスワークメモリ（115乃至117のメモリのうち一つ）にコピーして成功をリターンする（ステップ504、ステップ505）。

【0048】以上一連の処理が登録手段121の動作で

披露監視デバイスファイル (110~112) である場合

あり、エラーが発生するプロセスを特定する情報を基本
OSのワークエリアに予め記述しておくことにより、エ
ラ監視プロセスがエラー発生時にその情報を受け取り、エ
ラ一個所を認識できることになる。

【0049】次に、使用終了手段122について説明する。使用終了手段121は、エラー制御デバイスドライバ104の一部を構成するプログラムであり、いずれかのプロセスが終了してデバイスがクロールした時点で基本OSにより呼出され、そのプロセスに対応するデバイスファイルのワークメモリ(114~117)の状態フィールドをクロール状態にして、エラー発生プロセスを特定する情報を監視手段119の受信バッファにコピーすることにより、スリープ状態にあった監視手段119を実行状態にするものである。既に説明したように、この監視手段119によりエラー情報が監視プロセス106に通知されることでエラー検出が行われる。

【0050】監視プロセス106またはポート制御プロセス（107～109）が、それぞれ、オープン中の監視デバイスファイル110または被監視デバイスファイル（110～112）をクローズしたときに、基本OSはそのデバイスファイルのパラメータとして、エラー発生時のデバイスドライバ104の使用終了手段122を呼び出す。使用終了手段122は、クローズされたデバイスファイルが被監視デバイスファイル113かどうかを検出する（ステップ601）。

【0051】 クローズされたデバイスファイルが、監視
視デバイスファイル（110～112）でない（監視デ
バイスファイルである）場合は、監視デバイスワークメ
モリ 114 の状態フィールドが監視中であるか否かをチ
ェック（ステップ 610）、監視中でなければ成功を
リターンする（ステップ 611）。一方、監視中であれば
基本 OS のリブートインターフェースにより FAX 情
サービス装置 101 をリブートする（ステップ 61
2）。

【0052】このシステムリポート(ステップ612)が実行されるのは、監視中の監視プロセス106が何らかの障害により異常終了し、その障害基OSによってオゾン状態であった監視デバイスファイル113がグループ化された場合である。このように、監視中の監視プロセス106が異常終了したときに、エラー抑制デバイスドライバ104が装置をリセットする機能を受け、装置リポートの際に監視プロセスを自動的に立ち上げるよう設定することにより、監視プロセスの存在を定期的にウォッチする処理と、監視プロセスが異常終了した際のウオッチ処理を別途設けることなく、システムが制御不能な状態で放置されるという状況を防止することが可能となる。

【0053】次に、クローズされたデバイスファイルが
被監視デバイスファイル(110～112)である場合

について説明する。使用終了手段122は、デバイス番号のチェックを行い(ステップ602)、デバイス番号が不正であればエラー処理を行ない(ステップ603)、デバイス番号が正しければクローズされたデバイスソフトウェアに対応する監視デバイスソフトウェア(15から117のうちの1つ)の状態フィールドをクローズに設定し(ステップ604)、監視デバイスソフトウェア114の状態フィールドが監視中か否かをチェックする(ステップ605)。監視デバイスソフトウェア114の状態フィールドが監視中でない場合は、成功をリターンし(ステップ608)、監視中である場合は、監視デバイスソフトウェア114の受信バッファに、クローズされた監視デバイスソフトウェア(110～112)に対応する監視デバイスソフトウェア(115から117のうちの1つ)に格納されている情報(例えば、プロセスIDとポート番号)をコピーした後(ステップ606)、基本OSインターフェースであるWAKEUPをコールし(ステップ607)、受信待ちの状態になっている監視手段119を待ち状態から実行状態に変え、成功をリターンする(ステップ608)。

【0054】以上一連の処理が使用終了手段122の動作であるが、このように、監視デバイスソフトウェア(10～112)がクローズされるのは、ポート制御プロセス(107～109)がエラーを検知して前記デバイスソフトウェアをクローズする場合と、当該プロセスが例外等で異常終了し、基本OSがオーブン状態にある前記デバイスソフトウェアをクローズする場合と、がある。前者の場合には通常のエラー処理が実行されるが、後者のように、監視対象のポート制御プロセス(107～109)が異常終了した場合であっても、既に説明した監視手段により監視デバイスソフトウェア(115～117)に予め格納してある情報が監視デバイスソフトウェア113に必ず登録されるため、障害情報を確実に監視プロセスに通知することが可能となる。

【0055】従って、監視プロセスが定期的に監視プロセスの状態をチェックする必要がなくなり、エラー処理部のプログラム量を飛躍的に減らすことが可能になる。また、エラー検知にプロセス間通信を使用しないので装置の計算資源の消費を減らすことができる。

【0056】以上のようなソフトウェア構成をもつエラー制御デバイスドライバを使用して、監視プロセス106がポート制御プロセス(107～109)の状態監視、エラー処理を実行するフローを、図7を用いて具体的に説明する。この監視プロセス106は、FAX情報サービス装置101の起動、停止、ポート制御プロセス(107～109)の状態監視を行い、ポート制御プロセスが異常終了した場合、エラー処理として前記プロセスの再起動等を実行するようになっている。

【0057】まず、システムの起動処理が実行されると

(ステップ701)、ポート制御プロセス(107～109)を必要数(電話回線数)起動するとともに、監視デバイスソフトウェア113をオーブンして(ステップ702)、エラー監視用スレッドを生成する(ステップ703)。生成されたメインスレッドでは、ステップ712以下の監視プロセス106本来の処理、つまり、システム停止イベントの処理を実行する。イベント受信待ちの状態(ステップ712)、イベントを受信すると、その受信イベントがシステム停止か否かをチェックし(ステップ713)、それがシステム停止イベントでない場合にはイベントの処理(通常の処理)を実行して(ステップ714)、再びイベント受信待ちの状態に復帰する(ステップ712)。受信イベントがシステム停止イベントである場合には、監視停止手段を指定した入出力制御コールを実行して監視停止手段により監視処理を停止し(ステップ715)、システム停止処理を実行し(ステップ716)、監視プロセスを停止する(ステップ717)。

【0058】一方、生成されたサブスレッドでは以下の処理が実行される。エラーが発生した場合の障害情報を受け取るバッファアドレスをバグメータとして、監視手段を指定した入出力制御コールを発行して監視手段119を呼び出し(ステップ705)、エラーが発生するのを待つ。ポート制御プロセス(107～109)が異常終了した場合と監視プロセス106が監視停止手段120を呼び出した場合には、監視手段119は待ち状態から実行状態に移行し、ステップ705から処理が再開される。

【0059】次いで、監視手段119は重大なエラーが発生したのかを、フックシミュ通信ポートのハブドライバ等をアクセスしてチェックする(ステップ706)。重大エラーが発生した場合には装置の運用を停止するために、システム停止イベントをメインスレッドに送信し(ステップ707)、エラー監視スレッドを終了する(ステップ708)。

【0060】ステップ706でのチェックの結果、受信したエラーが重大エラーでないならば、監視停止手段が実行されたのかをチェックし(ステップ709)、監視停止が実行されたのであればスレッドを終了する(ステップ710)。また、監視停止が実行されたのではなく、受信したエラーが復旧可能なエラーである場合は、復旧処理を行った後(ステップ711)、ステップ705にジャンプしてエラー監視を続ける。復旧処理は、例えば、障害により制御プロセスが停止したポート番号を受信バッファから参照し、そのポートのポート制御プロセスを再起動することにより行なう。

【0061】以上一連の処理が監視プロセス106の動作であるが、このように監視プロセス106全体をメインスレッドとサブスレッドに分割し、通常のイベント処理とシステム停止処理とをメインスレッドで実行する一

方、監視手段119を呼び出してエラー検出をシエラリカバリーを実行する処理をサブスレッドで実行することにより、監視プロセス106がエラー監視をする処理としない処理とを明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を設ける必要がなくなる。更に、サブスレッドで監視手段119を呼び出す(ステップ705)のみでエラー監視が可能となり、プロセス間通信等を使用して定期的に監視プロセスの異常終了を検知する必要がなくなるため、エラー検知処理部が非常に簡素化される。

【0062】次に監視プロセスであるポート制御プロセス(107～109)の処理フローについて説明する。図8は、ポート制御プロセス(107～109)のエラー通知フローを示している。まず、制御する通信ポートの初期化処理を行い(ステップ801)、制御するポート番号を獲得して監視デバイスソフトウェア(110～112)をオーブンする(ステップ802)。次いで、登録すべき情報の一つであるプロセスIDを基本OSから獲得し(ステップ803)、入出力制御コールを基本OSに発行して登録手段を呼び出す(ステップ804)。更に、本来の処理であるフックシミュ通信処理を実行した後(ステップ805)、エラーチェックを行い(ステップ806)、エラーがない場合には、通常処理である通信処理(ステップ805)を繰り返す。

【0063】エラーを検知した場合には、監視デバイスソフトウェア(110～112)をクローズし(ステップ807)、ポート制御プロセス(107～109)を終了する(ステップ808)。このクローズ処理(ステップ807)により図6の使用終了手段122が呼び出され、ステップ804で登録手段121に登録しておいた情報が監視プロセスのバッファにコピーされ(ステップ806)、監視手段を実行状態に設定する(ステップ807)、これによりエラー情報が監視プロセス106に通知される。

【0064】以上一連の処理がポート制御プロセス(107～109)の動作であり、クローズ処理により障害情報を監視プロセスに通知するようにしたことにより、例外等が発生し監視プロセスが消滅した場合においても、基本OSがオーブンされている監視デバイスソフトウェアをクローズして、使用終了手段が確実にコールされ、監視プロセスに障害情報が確実に通知される。従って監視プロセスが監視プロセスの実行状況を定期的にプロセス間通信等により監視する必要がなくなり、エラー処理部が非常に簡単となり、さらに、計算資源の消費を抑えることができることとなる。

【0065】**【発明の効果】**以上の説明から明らかなように、本発明により、監視プロセスが定期的に監視プロセスの状態をチェックすることなく、障害簡易にエラー通知を行うことが可能となるため、エラー処理部のプログラム量を

飛躍的に減らすことが可能になり、また、エラー検知にプロセス間通信を使用しないので装置の計算資源の消費を減らすことができる。

【0066】更に、具体的には、エラー検出に必要な情報を監視プロセスとは異なる場所に保持しておき事後的に監視プロセスに通知することにより監視プロセスが異常終了した場合でも確実にエラー検出を行ない得る。また、システム起動時に際しては確実にエラーチェックの準備が整うことになり、監視プロセス起動直後にそのプロセスのエラー監視が可能になる。

【0067】また、エラー通知に必要な情報を監視プロセス終了時に確実に監視プロセスに通知することが可能となる。

【0068】また、監視手段がエラー検出を開始、続行するため、監視手段を呼び出すだけで全ての監視対象プロセスで生じたエラーを検知することが可能となる。

【0069】また、監視手段によりエラー監視を開始し、監視終了手段によりエラー監視を中断でき、監視不要な処理に対してのエラー監視、エラー通知を避けることが可能となる。

【0070】また、エラー監視をする処理としない処理を明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を実行する必要がなくなる。

【0071】また、監視プロセスが異常終了した場合でも確実にエラー検出、エラー通知を行ない得る。

【0072】また、監視プロセスの状態監視用のデバイスソフトウェアの利用によりエラー検知を確実に行ない得る。

【0073】更に、リセットにより必要となるプロセスを全て再生できるので、監視プロセスがなく装置の操作が不能になるという状況をなくすることが可能となる。

【図面の簡単な説明】

【図1】本発明のエラー制御デバイスドライバを使用したFAX情報サービス装置ソフトウェア構成図

【図2】本発明のエラー制御デバイスドライバの使用開始手段詳細フロー図

【図3】本発明のエラー制御デバイスドライバの監視手段詳細フロー図

【図4】本発明のエラー制御デバイスドライバの監視停止手段詳細フロー図

【図5】本発明のエラー制御デバイスドライバの登録手段詳細フロー図

【図6】本発明のエラー制御デバイスドライバの使用終了手段詳細フロー図

【図7】本発明の監視プロセスの処理フロー図

【図8】本発明のポート制御プロセスの処理フロー図

【図9】従来例によるエラー監視プロセスの詳細フロー図

【図10】従来例による監視プロセスの詳細フロー図

ロー図

【符号の説明】

104 エラー制御デバイスドライバ

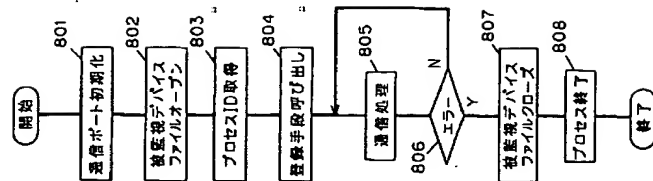
106 監視プロセス

107 ポート制御プロセス

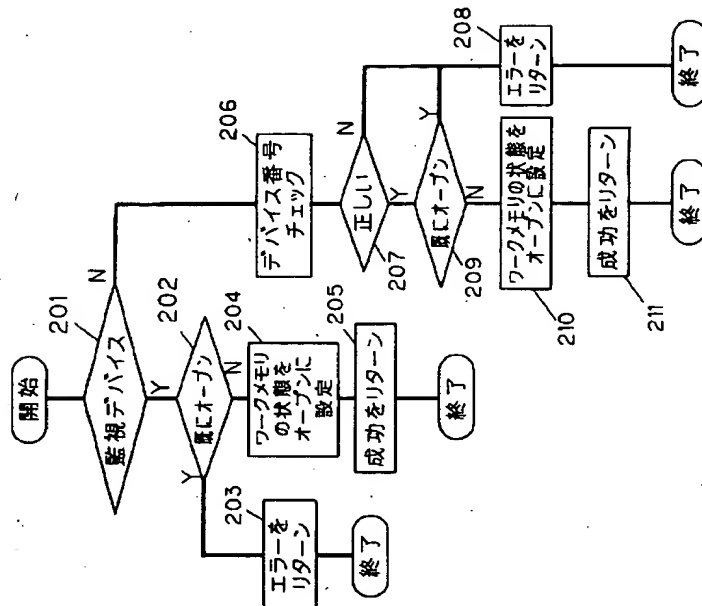
108 ポート制御プロセス

109 ポート制御プロセス

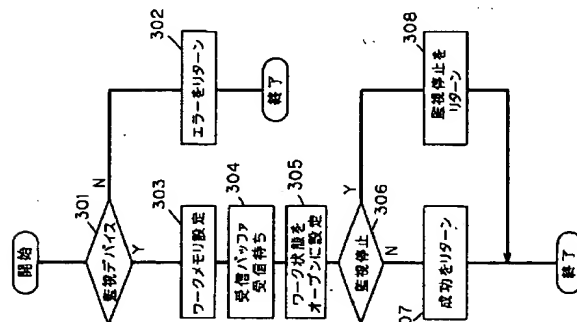
【図8】



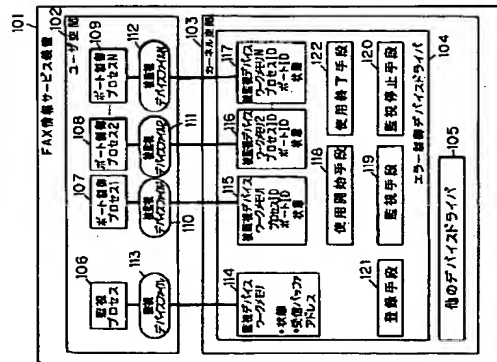
【図2】



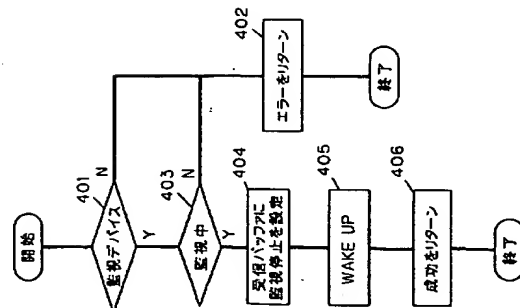
【図3】



【図11】



【図4】



【図6】

